

[MS-OXCSYNC]: Mailbox Synchronization Protocol Specification

Intellectual Property Rights Notice for Open Specifications Documentation

- **Technical Documentation.** Microsoft publishes Open Specifications documentation for protocols, file formats, languages, standards as well as overviews of the interaction among each of these technologies.
- **Copyrights.** This documentation is covered by Microsoft copyrights. Regardless of any other terms that are contained in the terms of use for the Microsoft website that hosts this documentation, you may make copies of it in order to develop implementations of the technologies described in the Open Specifications and may distribute portions of it in your implementations using these technologies or your documentation as necessary to properly document the implementation. You may also distribute in your implementation, with or without modification, any schema, IDL's, or code samples that are included in the documentation. This permission also applies to any documents that are referenced in the Open Specifications.
- **No Trade Secrets.** Microsoft does not claim any trade secret rights in this documentation.
- **Patents.** Microsoft has patents that may cover your implementations of the technologies described in the Open Specifications. Neither this notice nor Microsoft's delivery of the documentation grants any licenses under those or any other Microsoft patents. However, a given Open Specification may be covered by Microsoft's Open Specification Promise (available here: <http://www.microsoft.com/interop/osp>) or the Community Promise (available here: <http://www.microsoft.com/interop/cp/default.mspx>). If you would prefer a written license, or if the technologies described in the Open Specifications are not covered by the Open Specifications Promise or Community Promise, as applicable, patent licenses are available by contacting iplq@microsoft.com.
- **Trademarks.** The names of companies and products contained in this documentation may be covered by trademarks or similar intellectual property rights. This notice does not grant any licenses under those rights.
- **Fictitious Names.** The example companies, organizations, products, domain names, e-mail addresses, logos, people, places, and events depicted in this documentation are fictitious. No association with any real company, organization, product, domain name, email address, logo, person, place, or event is intended or should be inferred.

Reservation of Rights. All other rights are reserved, and this notice does not grant any rights other than specifically described above, whether by implication, estoppel, or otherwise.

Tools. The Open Specifications do not require the use of Microsoft programming tools or programming environments in order for you to develop an implementation. If you have access to Microsoft programming tools and environments you are free to take advantage of them. Certain Open Specifications are intended for use in conjunction with publicly available standard specifications and network programming art, and assumes that the reader either is familiar with the aforementioned material or has immediate access to it.

Revision Summary

Date	Revision History	Revision Class	Comments
04/04/2008	0.1		Initial Availability.
04/25/2008	0.2		Revised and updated property names and other technical content.
06/27/2008	1.0		Initial Release.
08/06/2008	1.01		Revised and edited technical content.
09/03/2008	1.02		Revised and edited technical content.
12/03/2008	1.03		Minor editorial fixes.
03/04/2009	1.04		Revised and edited technical content.
04/10/2009	2.0		Updated applicable product releases.
07/15/2009	3.0	Major	Revised and edited for technical content.
11/04/2009	4.0.0	Major	Updated and revised the technical content.
02/10/2010	4.1.0	Minor	Updated the technical content.
05/05/2010	4.1.1	Editorial	Revised and edited the technical content.
08/04/2010	4.2	Minor	Clarified the meaning of the technical content.
11/03/2010	5.0	Major	Significantly changed the technical content.
03/18/2011	6.0	Major	Significantly changed the technical content.
08/05/2011	6.0	No change	No changes to the meaning, language, or formatting of the technical content.
10/07/2011	7.0	Major	Significantly changed the technical content.
01/20/2012	7.0	No change	No changes to the meaning, language, or formatting of the technical content.

Table of Contents

1	Introduction	5
1.1	Glossary	5
1.2	References.....	6
1.2.1	Normative References.....	6
1.2.2	Informative References	6
1.3	Overview	6
1.4	Relationship to Other Protocols.....	8
1.5	Prerequisites/Preconditions	8
1.6	Applicability Statement.....	8
1.7	Versioning and Capability Negotiation.....	8
1.8	Vendor-Extensible Fields.....	8
1.9	Standards Assignments	8
2	Messages.....	9
2.1	Transport.....	9
2.2	Message Syntax	9
3	Protocol Details.....	10
3.1	Client Details.....	10
3.1.1	Abstract Data Model	10
3.1.1.1	Folder Scope	10
3.1.1.2	ICS State.....	10
3.1.1.3	Synchronization Upload and Download Contexts	10
3.1.1.4	Data Stream	10
3.1.1.5	Client Specific Handling	11
3.1.2	Timers	11
3.1.3	Initialization	11
3.1.3.1	Synchronization Setup Details	11
3.1.3.1.1	ID Reservation.....	11
3.1.3.1.1.1	Back-in-Time Detection	11
3.1.3.1.1.2	Mailbox Validation	11
3.1.3.2	Synchronization Configuration Details	12
3.1.3.2.1	Acquiring Synchronization Contexts	12
3.1.3.2.2	Synchronization Context Initialization	12
3.1.4	Higher-Layer Triggered Events.....	12
3.1.5	Message Processing Events and Sequencing Rules.....	13
3.1.5.1	Order of Operations	13
3.1.5.2	Upload Details.....	13
3.1.5.3	Download Details.....	13
3.1.5.3.1	Response Processing and Expected Sequencing	13
3.1.5.3.2	Property Deletes	13
3.1.5.3.3	Skip Bad Item.....	14
3.1.5.3.4	Partial Item Download	14
3.1.5.3.5	Client Headers	14
3.1.5.3.6	Client Header Completion.....	15
3.1.5.3.7	Throttling.....	15
3.1.5.3.8	Validation/Ignoring of Flags.....	15
3.1.5.3.9	ICS State Handling.....	15
3.1.5.4	Client Conflict Resolution	15
3.1.6	Timer Events	16

3.1.7 Other Local Events	16
3.1.7.1 Connectivity Loss.....	16
4 Protocol Examples.....	17
5 Security.....	18
5.1 Security Considerations for Implementers.....	18
5.2 Index of Security Parameters	18
6 Appendix A: Product Behavior.....	19
7 Change Tracking.....	20
8 Index	21

1 Introduction

The Mailbox Synchronization Protocol enables a client to synchronize **Message objects, Folder objects**, and their related properties with a server.

Sections 1.8, 2, and 3 of this specification are normative and contain RFC 2119 language. Section 1.5 and 1.9 are also normative but cannot contain RFC 2119 language. All other sections and examples in this specification are informative.

1.1 Glossary

The following terms are defined in [\[MS-GLOS\]](#):

flags
GUID
handle
property set
remote procedure call (RPC)

The following terms are defined in [\[MS-OXGLOS\]](#):

checkpoint ICS state
conflict resolution
content synchronization
EntryID
FastTransfer stream
final ICS state
Folder object
foreign identifier
hierarchy synchronization
Incremental Change Synchronization (ICS)
internal identifier
local replica
mailbox
Message object
messaging object
meta-property
offline
reminder
remote operation (ROP)
replica
ROP request
ROP request buffer
ROP response
ROP response buffer
store
stream
synchronization download context
synchronization upload context

The following terms are specific to this document:

MAY, SHOULD, MUST, SHOULD NOT, MUST NOT: These terms (in all caps) are used as described in [\[RFC2119\]](#). All statements of optional behavior use either MAY, SHOULD, or SHOULD NOT.

1.2 References

1.2.1 Normative References

We conduct frequent surveys of the normative references to assure their continued availability. If you have any issue with finding a normative reference, please contact dochelp@microsoft.com. We will assist you in finding the relevant information. Please check the archive site, <http://msdn2.microsoft.com/en-us/library/E4BD6494-06AD-4aed-9823-445E921C9624>, as an additional source.

[MS-OXCDATA] Microsoft Corporation, "[Data Structures](#)".

[MS-OXCFXICS] Microsoft Corporation, "[Bulk Data Transfer Protocol Specification](#)".

[MS-OXCMSG] Microsoft Corporation, "[Message and Attachment Object Protocol Specification](#)".

[MS-OXCROPS] Microsoft Corporation, "[Remote Operations \(ROP\) List and Encoding Protocol Specification](#)".

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997, <http://www.rfc-editor.org/rfc/rfc2119.txt>

1.2.2 Informative References

[MS-GLOS] Microsoft Corporation, "[Windows Protocols Master Glossary](#)".

[MS-OXCFOLD] Microsoft Corporation, "[Folder Object Protocol Specification](#)".

[MS-OXCRPC] Microsoft Corporation, "[Wire Format Protocol Specification](#)".

[MS-OXGLOS] Microsoft Corporation, "[Exchange Server Protocols Master Glossary](#)".

1.3 Overview

This protocol enables a client to keep synchronized versions of Message objects, Folder objects, and their related properties on both the client and the server. Changes that are made on the client are synchronized to the server and vice versa. Figure 1 illustrates the steps involved in synchronizing the data. In the case of **hierarchy synchronization**, the data being synchronized might be the properties of all descendant Folder objects, in the case of **content synchronization**, the data might be the Message objects contained within the folder being synchronized. Each box in the figure represents a logical state in the process of synchronizing the folder.

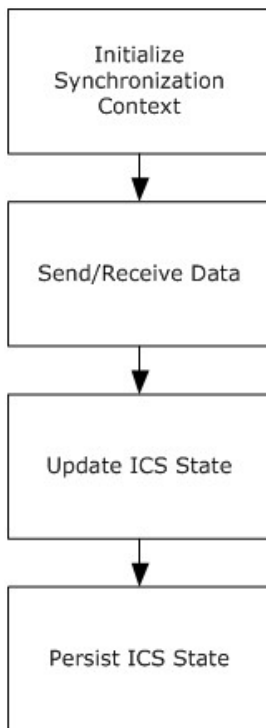


Figure 1: Synchronization states

The following list describes each state:

- Acquire Synchronization Context - To initiate the synchronization process, the client requests that the server acquire a **synchronization download context** and return the **handle** of the synchronization download context to the client.
- Initialize Synchronization Context - During initialization, the server determines the set of differences to transmit to the client. The client might send the server the **Incremental Change Synchronization (ICS)** state that was saved from a previous synchronization session so that the server only transmits differences since the last synchronization. If the client passes an empty ICS state when initializing a download, all **messaging object** data that is relevant to the folder being synchronized and the type of synchronization (hierarchy or content) is transmitted to the client.
- Send/Receive Data - By using the synchronization download context, data is now sent to or received from the server. The operations to accomplish this task depend on whether an upload or download synchronization is occurring.
- Update ICS State - When performing a download synchronization, the updated ICS state is sent to the client at the end of the **FastTransfer stream**. If the client wants to update the state in the middle of the download, a new synchronization download context has to be acquired from the server. Using this new context, the client downloads the ICS state from the server. When performing an upload, a **synchronization upload context** has to be acquired from the server. By using the synchronization upload context, the ICS state is downloaded from the server.
- Persist ICS State - By using whatever mechanism is prudent for the client to use, the ICS state is saved so that the client can use it to initialize the synchronization context. This is required because the server does not maintain per-client state. Note that if the client initializes the next

synchronization session without using the ICS state from the current session, the messaging object data that was processed in this synchronization session is retransmitted in the next session.

1.4 Relationship to Other Protocols

This protocol relies on an understanding of how to work with folders and messages as described in [\[MS-OXCMSG\]](#) and [\[MS-OXCFOLD\]](#). Additionally, this specification relies on an understanding of bulk data transfer, as described in [\[MS-OXCFXICS\]](#).

1.5 Prerequisites/Preconditions

This protocol assumes that the client previously acquired a handle to the folder for which it needs to synchronize data, as specified in [\[MS-OXCROPS\]](#) section 3.1.4.1.

1.6 Applicability Statement

This protocol is designed to detect folder and message changes from a known state. The client can store these changes in such a way as to constitute a **replica (2)** of the server data. The client might implement a replica (2) to facilitate creation, modification, or deletion of folders and messages while in an **offline** state.

1.7 Versioning and Capability Negotiation

The Bulk Data Transfer Protocol, as described in [\[MS-OXCFXICS\]](#), does not support detection of server capabilities. Instead, the client determines the server version to which it has connected. The client limits its behavior to the capabilities of the server version to which it has connected.

Several replication features were added for servers with a major version of eight (8). These features included partial item download and elimination of double-upload on send. For more information, see section [3.1.5.3.4](#) and [\[MS-OXCFXICS\]](#) section 3.3.4.3.3.2.1.2.

To determine the server version, inspect the version information that was returned when the connection to the server was established. For more information, see [\[MS-OXCRPC\]](#).

Additional capability negotiation information pertaining to synchronization is described in section [3.1.5.3.8](#) and [\[MS-OXCFXICS\]](#) section 1.7.

1.8 Vendor-Extensible Fields

None.

1.9 Standards Assignments

None.

2 Messages

The client accomplishes synchronization by using the **ROPs** defined in [\[MS-OXCFXICS\]](#) section 2.2.3. The following table lists the ROPs used in each stage of the upload and download synchronization process.

Operation	Initialization	Data transmission	Checkpointing
ICS download	RopSynchronizationConfigure ROP ([MS-OXCFXICS] section 2.2.3.2.1.1) RopSynchronizationUploadStateStreamBegin ROP ([MS-OXCFXICS] section 2.2.3.2.2.1) RopSynchronizationUploadStateStreamContinue ROP ([MS-OXCFXICS] section 2.2.3.2.2.2) RopSynchronizationUploadStateStreamEnd ROP ([MS-OXCFXICS] section 2.2.3.2.2.3)	RopFastTransferSourceGetBuffer ROP ([MS-OXCFXICS] section 2.2.3.1.1.5)	RopSynchronizationGetTransferState ([MS-OXCFXICS] section 2.2.3.2.3.1) and the RopFastTransferSourceGetBuffer ROPs ([MS-OXCFXICS] section 2.2.3.1.1.5) MAY <1> be used
ICS upload	RopSynchronizationOpenCollector ROP ([MS-OXCFXICS] section 2.2.3.2.4.1) RopSynchronizationUploadStateStreamBegin ROP ([MS-OXCFXICS] section 2.2.3.2.2.1) RopSynchronizationUploadStateStreamContinue ROP ([MS-OXCFXICS] section 2.2.3.2.2.2) RopSynchronizationUploadStateStreamEnd ROP ([MS-OXCFXICS] section 2.2.3.2.2.3)	RopSynchronizationImportMessageChange ROP ([MS-OXCFXICS] section 2.2.3.2.4.2) RopSynchronizationImportHierarchyChange ROP ([MS-OXCFXICS] section 2.2.3.2.4.3) RopSynchronizationImportMessageMove ROP ([MS-OXCFXICS] section 2.2.3.2.4.4) RopSynchronizationImportDeletes ROP ([MS-OXCFXICS] section 2.2.3.2.4.5) RopSynchronizationImportReadStateChanges ROP ([MS-OXCFXICS] section 2.2.3.2.4.6)	RopSynchronizationGetTransferState ROP ([MS-OXCFXICS] section 2.2.3.2.3.1) RopFastTransferSourceGetBuffer ROP ([MS-OXCFXICS] section 2.2.3.1.1.5)

2.1 Transport

The **ROP request buffers** and **ROP response buffers** utilized by this protocol are sent to and received by the server by using the underlying Remote Operations (ROP) List and Encoding Protocol, as specified in [\[MS-OXCROPS\]](#).

2.2 Message Syntax

This protocol relies on the ROPs defined in [\[MS-OXCFXICS\]](#) section 2.2.3 to synchronize data.

3 Protocol Details

When participating in synchronization, the client has several responsibilities in addition to the actual act of synchronization. These include: ID assignment, change tracking, **conflict resolution**, and ICS state storage.

3.1 Client Details

3.1.1 Abstract Data Model

This section describes a conceptual model of possible data organization that an implementation maintains to participate in this protocol. The described organization is provided to facilitate the explanation of how the protocol behaves. This specification does not mandate that implementations adhere to this model as long as their external behavior is consistent with that described.

3.1.1.1 Folder Scope

The synchronization portions of the Bulk Data Transfer Protocol, which specifies the messages used by this protocol, are fundamentally associated with a folder. To initiate a synchronization action, the client first acquires a handle to a folder with which to synchronize. When performing a content synchronization, only the messages contained in that folder are synchronized. When performing a hierarchy synchronization, all descendants of that folder are synchronized. Hierarchy synchronization only synchronizes the folder structure of the hierarchy and the properties of the folders. The messages contained in the folder are not synchronized when performing a hierarchy synchronization.

3.1.1.2 ICS State

The ICS state is a logical bookmark describing the point at which the last synchronization completed. This state is presented to a server in a subsequent synchronization session, enabling only changes made since that state was saved to be sent to the client. If the ICS state is not saved by the client, then all the messages in the folder at the time of synchronization are sent to the client. Therefore, it is the responsibility of the client to save this state at the end of each synchronization.

3.1.1.3 Synchronization Upload and Download Contexts

Synchronization upload contexts and synchronization download contexts are the basis for all synchronization. A synchronization upload context is obtained from the server to perform the upload of changes. Conversely, synchronization download contexts are used to initiate the download of server changes to the client.

3.1.1.4 Data Stream

The synchronization portions of the Bulk Data Transfer Protocol are extensions to the core FastTransfer portions of the Bulk Data Transfer Protocol, as specified in [\[MS-OXCFXICS\]](#). Upon configuration of a synchronization download context (and subsequent initialization using the persisted ICS state), the client retrieves the FastTransfer stream containing all the changes. This FastTransfer stream is parsed according to [\[MS-OXCFXICS\]](#) section 2.2.4 in order to see the message and folder changes at the object level.

3.1.1.5 Client Specific Handling

Clients can choose to handle pieces of the data **stream** in specific ways to improve the efficiency or user experience. Examples of this from [\[MS-OXCFCICS\]](#) are progress information, as specified in [\[MS-OXCFCICS\]](#) section 2.2.2.7, and bad item notifications. A possible implementation uses progress information to compute and inform the end user how much time is left in the synchronization session. Bad item notifications can be used to move the problem items to another folder to avoid repeated errors on subsequent synchronization sessions.

3.1.2 Timers

None.

3.1.3 Initialization

3.1.3.1 Synchronization Setup Details

3.1.3.1.1 ID Reservation

When items are created in the **local replica**, IDs MUST be assigned in a way that is compatible with the server. It is acceptable for the underlying storage to assign a **foreign identifier** that is only for local use. However, all communications with the server MUST be based on server-compatible **internal identifiers**. To facilitate this, a client MUST reserve a set of internal identifiers and use those when new items are created locally. This is accomplished by using the **RopGetLocalReplicaIds** ROP ([\[MS-OXCROPS\]](#) section 2.2.13.13). Note that each time this call is made, ID ranges on the server are consumed. The client SHOULD NOT use the **RopGetLocalReplicaIds** ROP until the bulk of the previously reserved IDs are consumed. If the **RopGetLocalReplicaIds** ROP is called prematurely, resources are unnecessarily consumed on the server.

The internal identifiers received can be used for both folders and messages. The internal identifiers MUST be consumed by adding one to each value of the **GlobalCount** field from the **RopGetLocalReplicaIds** ROP. The client MUST NOT reuse IDs. To avoid fragmenting the IDSET, clients SHOULD use contiguous subranges for items within the same folder.

A client MAY use the **RopSetLocalReplicaMidsetDeleted** ROP ([\[MS-OXCROPS\]](#) section 2.2.13.12) to inform the server of IDs that were previously reserved that the client will not be using in a given folder.

3.1.3.1.1.1 Back-in-Time Detection

Due to the ID reservation and the fact that a client MUST NOT reuse IDs, the server MUST implement some mechanism to detect client rollback. One possible source of rollback is restoring a local replica from a backup after that local replica was used past the point in time when it was backed up.

A sample implementation is a property on both the local replica and the server that stores a counter specific to each replica (2). When the replica (2) connects to the server, it verifies that the counter is greater than or equal to the server counter. If the client counter is ever less than the server counter, rollback has occurred and that client replica (2) is abandoned.

3.1.3.1.1.2 Mailbox Validation

A client MUST NOT let a replica (2) of one **mailbox** synchronize with a different mailbox. For this reason, the client MUST identify the mailbox that belongs to any given replica (2).

This can be accomplished by using the mailbox instance **GUID**, which is returned by the **RopLogon** ROP ([\[MS-OXCROPS\]](#) section 2.2.3.1). The mailbox is associated with the local replica and compared to the mailbox GUID after the **RopLogon** ROP completed.

3.1.3.2 Synchronization Configuration Details

3.1.3.2.1 Acquiring Synchronization Contexts

If an upload synchronization is being performed, a handle to the synchronization upload context MUST be obtained by using the **RopSynchronizationOpenCollector** ROP ([\[MS-OXCROPS\]](#) section 2.2.13.7). If a download synchronization is being performed, a handle to the synchronization download context MUST be obtained by using the **RopSynchronizationConfigure** ROP ([\[MS-OXCROPS\]](#) section 2.2.13.1).

3.1.3.2.2 Synchronization Context Initialization

First, the client MUST use the persisted ICS state. If this is a first-time synchronization, the client MUST pass an empty ICS state.

To initialize a synchronization download context, the handle to the context MUST be used to pass the ICS state up to the server by using the **RopSynchronizationUploadStateStreamBegin** ROP ([\[MS-OXCROPS\]](#) section 2.2.13.9). Next, use the **RopSynchronizationUploadStateStreamContinue** ROP ([\[MS-OXCROPS\]](#) section 2.2.13.10). Finally, after the ICS state upload is complete, the **RopSynchronizationUploadStateStreamEnd** ROP ([\[MS-OXCROPS\]](#) section 2.2.13.11) MUST be used.

For a download synchronization, initialization MUST occur before the **RopFastTransferSourceGetBuffer** ROP ([\[MS-OXCROPS\]](#) section 2.2.12.3) (or any other ICS ROP) can be issued.

3.1.4 Higher-Layer Triggered Events

The following events apply to synchronization:

- Initialization
- Cancellation

Initialization occurs when a client initializes a synchronization session. This can occur as the result of user action, an application-specific timer, or a response to a notification from the server.

Cancellation occurs as a result of the user terminating the client or requesting that synchronization operations stop (for a transition to offline state, for example) before the client successfully committed all items in the buffer.

When a cancellation occurs on a download, the client SHOULD NOT update the ICS state by using the **RopSynchronizationGetTransferState** ROP ([\[MS-OXCROPS\]](#) section 2.2.13.8) if there is any data left unprocessed from the result of the **RopFastTransferSourceGetBuffer** ROP ([\[MS-OXCROPS\]](#) section 2.2.12.3). This is because the server does not know if you successfully committed all items. It is possible to miss items if the **RopSynchronizationGetTransferState** ROP is used at a time when synchronization buffers are only partially processed.

Because the state cannot be updated safely, the items that were downloaded during the canceled synchronization session are be downloaded again during the next download. To reduce the number of items that need to be downloaded again to the client, use checkpointing, as specified in [\[MS-OXCFICS\]](#) section 3.3.5.6.

For more details about persisting the state when the **RopFastTransferSourceGetBuffer** ROP completed successfully, see section [3.1.5.3.9](#).

3.1.5 Message Processing Events and Sequencing Rules

3.1.5.1 Order of Operations

When performing synchronization, a client SHOULD perform upload before download. This allows conflicts to be resolved before data is received by the client. In this way, wire efficiency can be increased, as an additional post-resolution upload is not required.

3.1.5.2 Upload Details

Client upload details are specified in [\[MS-OXCFXICS\]](#) section 3.3.4.3.3.

3.1.5.3 Download Details

When performing a download, a client MUST use the **RopFastTransferSourceGetBuffer** ROP ([\[MS-OXCROPS\]](#) section 2.2.12.3). The response contains a FastTransfer stream as previously mentioned in section [3.1.1.4](#).

The following diagram shows the download details.

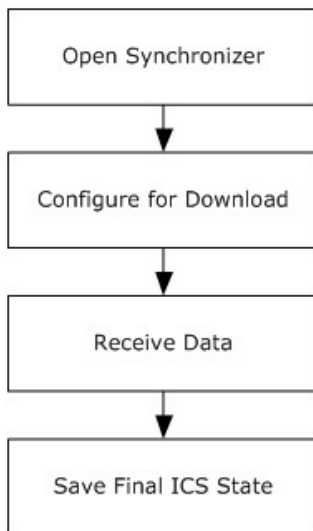


Figure 2: Download details

3.1.5.3.1 Response Processing and Expected Sequencing

Response processing occurs as specified in [\[MS-OXCFXICS\]](#). There are several best practices for dealing with particular ICS markers, as specified in [\[MS-OXCFXICS\]](#) section 2.2.4.1.4. These are specified in the following sections.

3.1.5.3.2 Property Deletes

When a property is deleted, clients MUST remove all properties on a folder or message and replace them with the complete set of downloaded properties supplied by the server. Or, if the client and

server support partial item downloads, as specified in [\[MS-OXCFXICS\]](#) section 1.7, all the properties in the property group MUST be removed and replaced. For more details about partial item downloads, see section [3.1.5.3.4](#) and [\[MS-OXCFXICS\]](#) section 3.2.5.7. Failure to comply risks leaving items in an inconsistent state.

3.1.5.3.3 Skip Bad Item

The client SHOULD set the **RecoverMode** flag of the **SendOptions** field in the **RopSynchronizationConfigure** ROP request ([\[MS-OXCROPS\]](#) section 2.2.13.1), in order to receive the **errorInfo** element, as specified in [\[MS-OXCFXICS\]](#) section 2.2.4.3.4, and the **FxErrorInfo** marker, as specified in [\[MS-OXCFXICS\]](#) section 2.2.4.1.4, in the FastTransfer stream when the server encounters an error processing an item for download. Using the folder ID (FID) and message ID (MID) following the **FxErrorInfo** marker, the client SHOULD move the item into a folder that is not synchronized. In this way, the error does not continue to occur during each subsequent synchronization.

3.1.5.3.4 Partial Item Download

Users typically modify items in relatively minor ways. Flagging, adding **reminders**, or adding categories are common ways in which users modify items. In order to efficiently download these changes to the client, this protocol uses partial item downloads.

This method of download groups message properties into property groups that are defined by the server. When processing the FastTransfer stream, the client will observe multiple sets of **PropertyGroup** structures, as specified in [\[MS-OXCFXICS\]](#) section 2.2.2.8.1, as part of the **groupInfo** element, as specified in [\[MS-OXCFXICS\]](#) section 2.2.4.3.8. There might be multiple property groups, as the server is free to define new groups at any time. Sometime later in the FastTransfer stream, the client observes the **MetaTagIncrementalSyncMessagePartial meta-property**, as specified in [\[MS-OXCFXICS\]](#) section 2.2.4.1.5.5, which is the index of the set of property groups that applies for the subsequent messages. This gives the client enough information to handle partial items. While processing the FastTransfer stream for message changes, the client can receive the **IncrSyncChgPartial** marker, as specified in [\[MS-OXCFXICS\]](#) section 2.2.4.1.4, which informs the client that they are about to receive a message header for a partial change. After the header has been received, the client receives the partial message data, as specified by [\[MS-OXCFXICS\]](#).

The client MUST pass the **PartialItem** flag of the **SendOptions** field, as specified in [\[MS-OXCFXICS\]](#) section 2.2.3.1.1.1.1, in the **RopSynchronizationConfigure ROP request** ([\[MS-OXCROPS\]](#) section 2.2.13.1) to indicate to the server that it supports partial item downloads.

When processing a partial change, the client MUST NOT delete all properties on the local item, as specified in section [3.1.5.3.2](#). Instead, the client MUST only delete the properties on the local item that are defined in the group that is specified by the server. For more details about server behavior related to partial item downloads, see [\[MS-OXCFXICS\]](#) section 3.2.5.7.

3.1.5.3.5 Client Headers

It is often desirable to avoid downloading a message in its entirety when just a few properties are sufficient for the user (for example, to, from, subject, received time). This is due to the fact that the user might have a slow connection or might delete most of their mail without reading it. The Bulk Data Transfer Protocol, as specified in [\[MS-OXCFXICS\]](#), has a provision for this type of synchronization. To do this, the client MUST include the **property set** and specify the **OnlySpecifiedProperties** flag of the **SynchronizationFlag** field in the **RopSynchronizationConfigure** ROP request ([\[MS-OXCROPS\]](#) section 2.2.13.1).

3.1.5.3.6 Client Header Completion

When clients download client headers, they often have to turn those client headers into full items. To do this, a client MUST use the **RopFastTransferSourceCopyMessages** ROP ([\[MS-OXCROPS\]](#) section 2.2.12.5). In addition, the client MUST pass the **SendEntryId** flag of the **CopyFlags** field, as specified in [\[MS-OXCFXICS\]](#) section 2.2.3.1.1.3.1, in the **RopFastTransferSourceCopyMessages** ROP request. This tells the server to pass the original **EntryID** of the item to the client. The messages come down in the FastTransfer stream with an additional property, **PidTagOriginalEntryId** ([\[MS-OXCFXICS\]](#) section 2.2.1.2.9), containing the EntryID. This is required because the messages downloaded via the **RopFastTransferSourceCopyMessages** ROP are considered to be message copies and the FastTransfer stream contains no correlating identification properties. The **PidTagOriginalEntryId** property is one way the client can associate the full message item being downloaded from the server with the message header stored in the client.

3.1.5.3.7 Throttling

If the client receives a **RopFastTransferSourceGetBuffer** ROP response, as specified in [\[MS-OXCFXICS\]](#) section 2.2.3.1.1.5, where the value of the **ReturnValue** field is ServerBusy (0x8004010B), as specified in [\[MS-OXCDATA\]](#) section 2.4, the client SHOULD wait before sending another **RopFastTransferSourceGetBuffer** ROP request. The amount of time to wait is specified by the **BackoffTime** field in the **RopFastTransferSourceGetBuffer** ROP response, in milliseconds. Failure to wait can result in the client receiving additional failure responses.

3.1.5.3.8 Validation/Ignoring of Flags

Clients that have not performed version detection on the server and have leveraged the server's flexibility to ignore unknown **flags** in the **SynchronizationExtraFlag** field of the **RopSynchronizationConfigure** ROP ([\[MS-OXCROPS\]](#) section 2.2.13.1) MUST parse the FastTransfer stream to detect whether the server honored each of the requested behavior flags. This enables the client to request behavior from an older server that does not support newer behaviors without having to strictly check the version of the server.

3.1.5.3.9 ICS State Handling

After an upload finishes, the client SHOULD send a **RopSynchronizationGetTransferState** ROP request ([\[MS-OXCROPS\]](#) section 2.2.13.8) to retrieve the **final ICS state**.

When a download successfully finishes, the final ICS state is included in the last buffer or buffers returned from the **RopFastTransferSourceGetBuffer** ROP response, as specified in [\[MS-OXCFXICS\]](#) section 2.2.3.1.1.5.2.

The client SHOULD persist this state in the local data **store** and use it upon initializing the next synchronization action. Failure to do so results in all items being returned to the client on the next download synchronization.

For more details about uploading the ICS state, see [\[MS-OXCFXICS\]](#) section 3.3.4.3.1. For more details about downloading the ICS state, see [\[MS-OXCFXICS\]](#) section 3.3.4.3.4.

3.1.5.4 Client Conflict Resolution

A client can try to automatically perform conflict resolution as specified in [\[MS-OXCFXICS\]](#) section 3.3.5.11, thereby avoiding the creation of a conflict resolve message.

When a client is unable to automatically resolve all conflicting changes, it SHOULD have a strategy to preserve alternate versions of messages, which SHOULD be made accessible to the user in case they prefer the alternate version.

A client can, for example, pick a "winner" message based on the **PidTagLastModificationTime** property ([\[MS-OXCMSG\]](#) section 2.2.2.2) and leave this in place of the modified item. The other version of the message, deemed the "loser," can be moved to a folder that contains previous versions of conflicting messages. If a strategy similar to this is implemented, these items SHOULD be linked by using the **PidTagConflictItems** property.

3.1.6 Timer Events

None.

3.1.7 Other Local Events

3.1.7.1 Connectivity Loss

If the connection is lost or any other recoverable error takes place, it is not possible to update the ICS state. Because the state cannot be updated, the items that were downloaded by the aborted **RopFastTransferSourceGetBuffer** ROP ([\[MS-OXCROPS\]](#) section 2.2.12.3) are re-downloaded the next time the **RopFastTransferSourceGetBuffer** ROP is called. To reduce the number of items that need to be downloaded again the client can use checkpointing, as specified in [\[MS-OXCFXICS\]](#) section 3.3.5.6.

4 Protocol Examples

For an example showing the ROPs involved in synchronizing changes between the client and server when adding or modifying a folder, see [\[MS-OXCFXICS\]](#) section 4.1.

For an example showing the ROPs involved in synchronizing changes between the client and server when creating or modifying a message, see [\[MS-OXCFXICS\]](#) section 4.2.

For an example showing the ROPs involved in synchronizing changes between the client and server when uploading a partial message to the server, see [\[MS-OXCFXICS\]](#) section 4.3.

5 Security

5.1 Security Considerations for Implementers

There are no special security considerations specific to this protocol. General security considerations that pertain to the underlying **RPC**-based transport apply, as described in [\[MS-OXCROPS\]](#).

5.2 Index of Security Parameters

None.

6 Appendix A: Product Behavior

The information in this specification is applicable to the following Microsoft products or supplemental software. References to product versions include released service packs:

- Microsoft® Exchange Server 2003
- Microsoft® Exchange Server 2007
- Microsoft® Exchange Server 2010
- Microsoft® Office Outlook® 2003
- Microsoft® Office Outlook® 2007
- Microsoft® Outlook® 2010

Exceptions, if any, are noted below. If a service pack or Quick Fix Engineering (QFE) number appears with the product version, behavior changed in that service pack or QFE. The new behavior also applies to subsequent service packs of the product unless otherwise specified. If a product edition appears with the product version, behavior is different in that product edition.

Unless otherwise specified, any statement of optional behavior in this specification that is prescribed using the terms SHOULD or SHOULD NOT implies product behavior in accordance with the SHOULD or SHOULD NOT prescription. Unless otherwise specified, the term MAY implies that the product does not follow the prescription.

[<1> Section 2:](#) In Exchange 2003 and Exchange 2007, the **RopSynchronizationGetTransferState** ([\[MS-OXCROPS\]](#) section 2.2.13.8) and the **RopFastTransferSourceGetBuffer** ([\[MS-OXCROPS\]](#) section 2.2.12.3) ROPs are used to checkpoint ICS download operations. In Exchange 2010, trying to retrieve a **checkpoint ICS state** during the download returns the initial state, and not a state with differences applied.

7 Change Tracking

No table of changes is available. The document is either new or has had no changes since its last release.

8 Index

A

Abstract data model
[client](#) 10
[Applicability](#) 8

C

[Capability negotiation](#) 8
[Change tracking](#) 20
Client
 [abstract data model](#) 10
 [higher-layer triggered events](#) 12
 [timer events](#) 16
 [timers](#) 11

D

Data model - abstract
 [client](#) 10

F

[Fields - vendor-extensible](#) 8

G

[Glossary](#) 5

H

Higher-layer triggered events
 [client](#) 12

I

[Implementer - security considerations](#) 18
[Index of security parameters](#) 18
[Informative references](#) 6
[Introduction](#) 5

M

Messages
 [transport](#) 9

N

[Normative references](#) 6

O

[Overview \(synopsis\)](#) 6

P

[Parameters - security index](#) 18
[Preconditions](#) 8
[Prerequisites](#) 8

[Product behavior](#) 19

R

References
 [informative](#) 6
 [normative](#) 6
[Relationship to other protocols](#) 8

S

Security
 [implementer considerations](#) 18
 [parameter index](#) 18
[Standards assignments](#) 8

T

Timer events
 [client](#) 16
Timers
 [client](#) 11
[Tracking changes](#) 20
[Transport](#) 9
Triggered events - higher-layer
 [client](#) 12

V

[Vendor-extensible fields](#) 8
[Versioning](#) 8